AIMLZG537
Information Retrieval

Dr. Maheswari Karthikeyan

Lecture2:01-08-2026

BITS Pilani
Pilani Campus

Lecture Outline

innovate achieve lead

Boolean Retrieval

- Inverted Index
- Processing Boolean queries

Slides are from Introduction to Information Retrieval;Manning,Raghavan

http://nlp.stanford.edu/IR-book

An Example of IR problem

innovate achieve lead

- Example: Shakespeare

  - Find all Shakespeare plays that contain the words

BRUTUS AND CAESAR AND NOT CALPURNIA

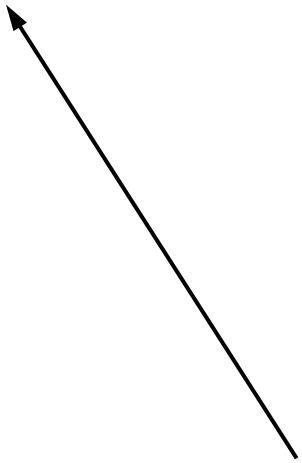
An Example of IR problem

innovate achieve lead

- One solution is Linear Scan of Document (Brute Force Approach)
- One could grep all of Shakespeare's plays for BRUTUS and CAESAR, then strip out plays containing CALPURNIA
  -- Similar to Unix command grep
- But this may not be the solution, because….
  - Slow for large data collection (e.g., the web, which contains billions or trillions of words)
  - Does not allow flexible matching operations (can't search Brutus and Caesar in the same sentence)
  - Ranked retrieval (best documents to return)
- **Instead, organize beforehand (Index the Documents)**

Term-Document Incidence Matrix

- Two dimensional: Terms and Documents
- Matrix element (t, d) = 1 if term t appears in document d



1 if play contains word, 0 otherwise

Brutus AND Caesar but NOT Calpurnia

Query Answer

- Take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** (complemented) □ bitwise *AND*.

110100 AND 110111 AND NOT 010000

110100 AND 110111 AND 101111 = 100100

Ans : Antony and Cleopatra, Hamlet

Query Answer

Query Answer

innovate achieve lead

Query Results implies…..

- Using the term vectors, we can only find whether the documents meet the query, but cannot find which parts of the documents meet the query.

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius ***Caesar*** dead,
He cried almost to roaring; and he wept
When at Philippi he found ***Brutus*** slain.

Hamlet, Act III, Scene ii

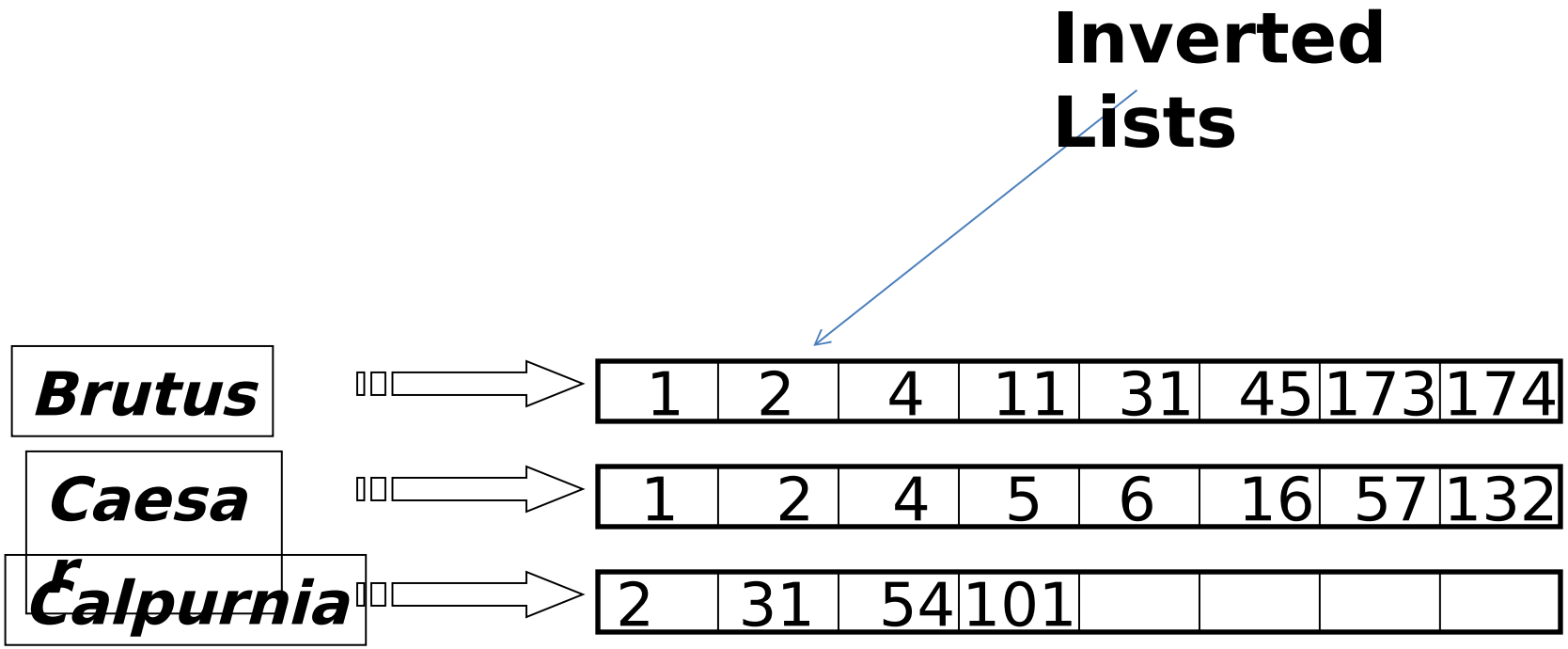
Lord Polonius: I did enact Julius ***Caesar*** I was killed i' the
Capitol; ***Brutus*** killed me.

Term-Document Incidence Matrix

innovate achieve lead

- The incidence matrix is sparse
- Term Vectors are of fixed length **(size?)**
- Difficult to build for too big Document Corpus
- **Solution is to record only the things that occur.(Inverted Index)**

Inverted Index

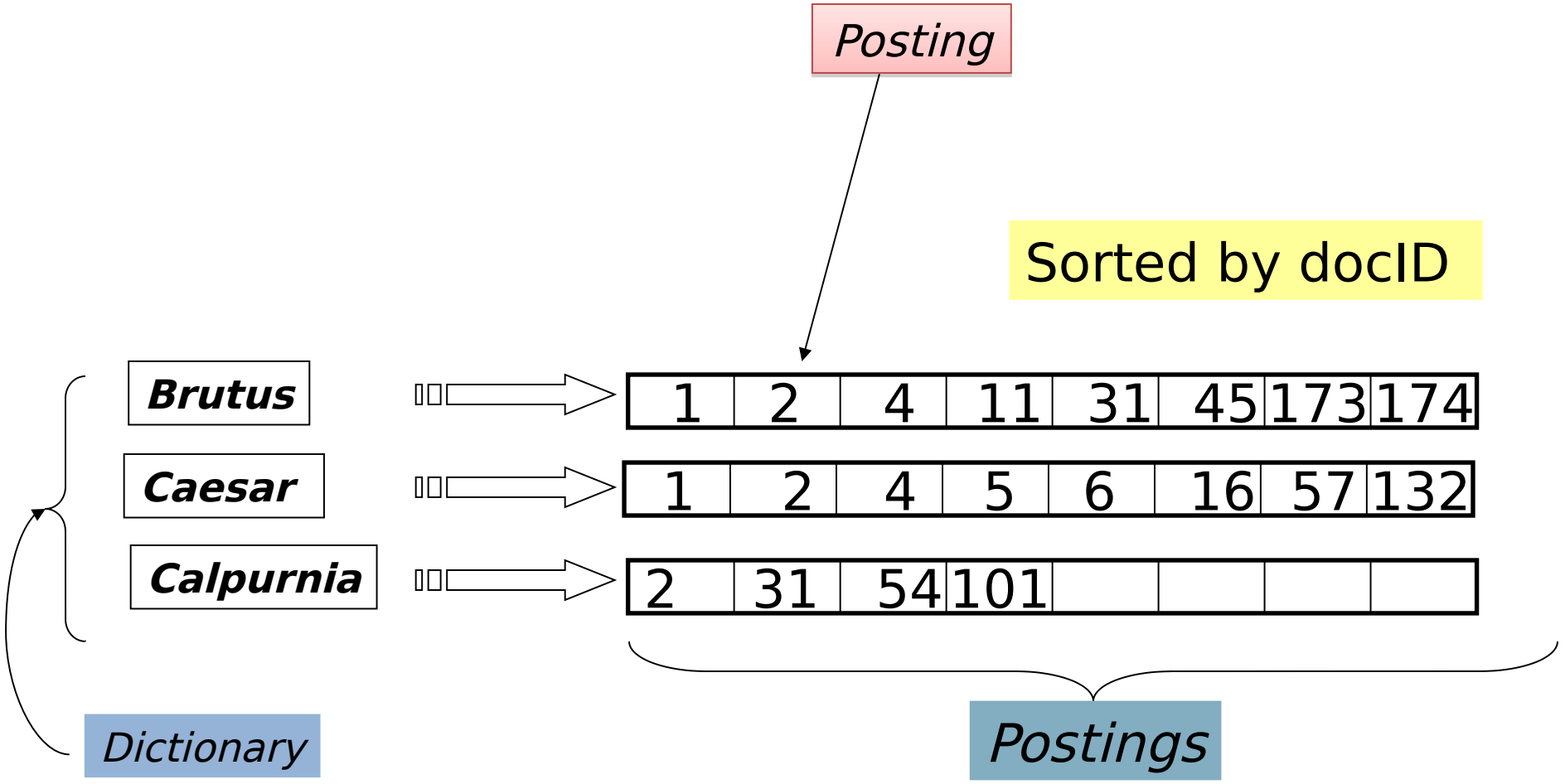


Inverted Index

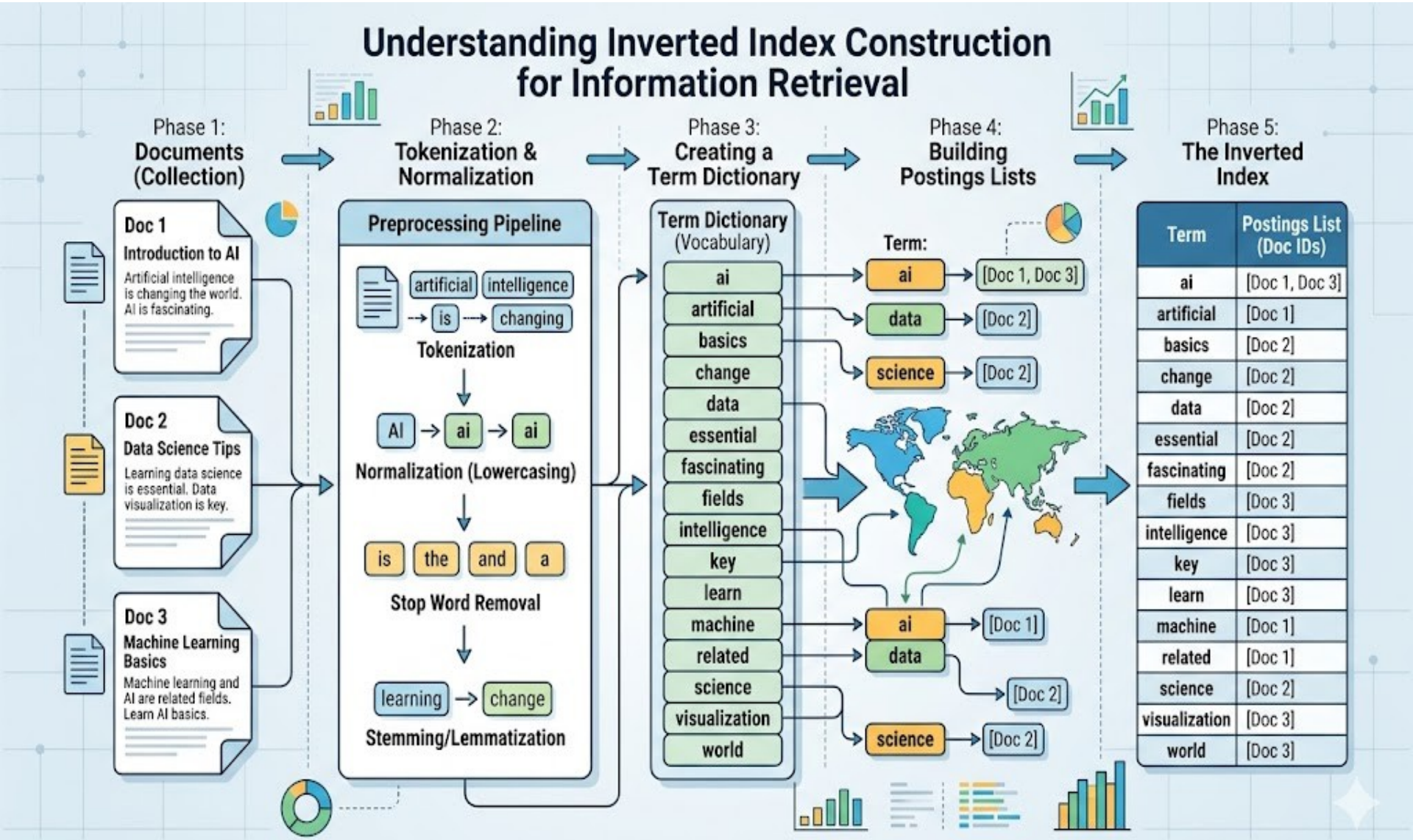
Building an Inverted Index

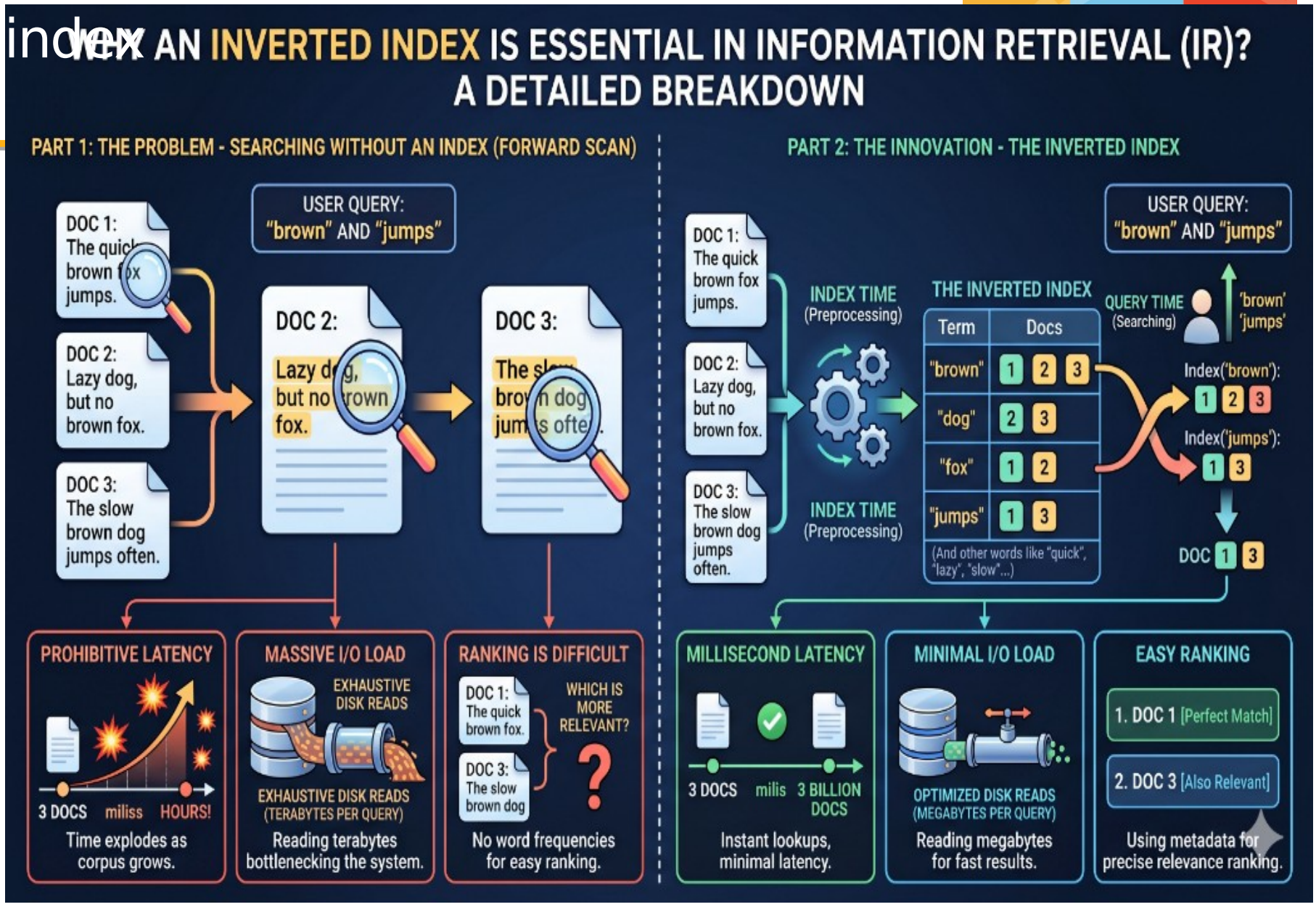
- For each term *t*, we must store a list of all documents that contain *t*.
  - Identify each by a **docID**, a document serial number
- Sort according to **docID**
- Instances of the same term are grouped and split into a dictionary and posting
- Can be implemented using either singly linked lists or variable length arrays

Inverted Index



Index construction




WHY AN INVERTED INDEX IS ESSENTIAL IN INFORMATION RETRIEVAL (IR)?
A DETAILED BREAKDOWN
PART 1: THE PROBLEM - SEARCHING WITHOUT AN INDEX (FORWARD SCAN)
USER QUERY:
"brown" AND "jumps"
DOC 1:
The quick brown fox jumps.
DOC 2:
Lazy dog, but no brown fox.
DOC 3:
The slow brown dog jumps often.
PROHIBITIVE LATENCY
3 DOCS
miliss
HOURS!
Time explodes as corpus grows.
MASSIVE I/O LOAD
EXHAUSTIVE DISK READS
EXHAUSTIVE DISK READS (TERABYTES PER QUERY)
Reading terabytes bottlenecking the system.
RANKING IS DIFFICULT
DOC 1:
The quick brown fox.
DOC 3:
The slow brown dog
WHICH IS MORE RELEVANT?
No word frequencies for easy ranking.
PART 2: THE INNOVATION - THE INVERTED INDEX
USER QUERY:
"brown" AND "jumps"
DOC 1:
The quick brown fox jumps.
DOC 2:
Lazy dog, but no brown fox.
DOC 3:
The slow brown dog jumps often.
INDEX TIME
(Preprocessing)
INDEX TIME
(Preprocessing)
THE INVERTED INDEX
Term
Docs
"brown" 1 2 3
"dog" 2 3
"fox" 1 2
"jumps" 1 3
(And other words like "quick", "lazy", "slow"...)
QUERY TIME
(Searching)
'brown'
'jumps'
Index('brown'): 1 2 3
Index('jumps'): 1 3
DOC 1 3
MILLISECOND LATENCY
3 DOCS
milis
3 BILLION DOCS
Instant lookups, minimal latency.
MINIMAL I/O LOAD
OPTIMIZED DISK READS (MEGABYTES PER QUERY)
Reading megabytes for fast results.
EASY RANKING
1. DOC 1 [Perfect Match]
2. DOC 3 [Also Relevant]
Using metadata for precise relevance ranking.

Building Inverted Index

- Collect the documents to be indexed:

Friends, Romans, countrymen.

- Tokenize the text, turning each document into a list of tokens:

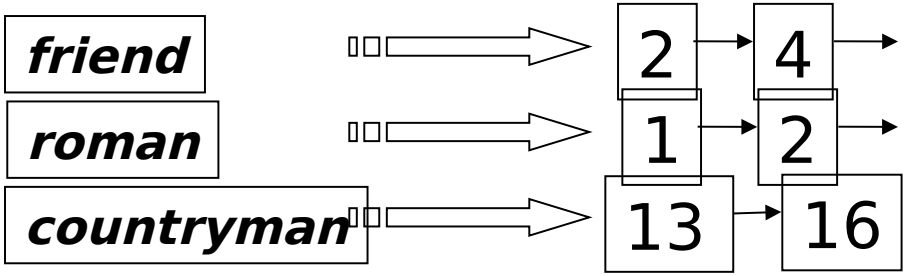
Friends	Romans	Countrymen

Building Inverted Index

- Do linguistic pre-processing, producing a list of normalized tokens, which are the indexing terms

friend roman countryman

- Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.



Building Inverted Index

Sequence of (Modified token, Document ID) pairs.

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Building Inverted Index

innovate achieve lead

Sort by

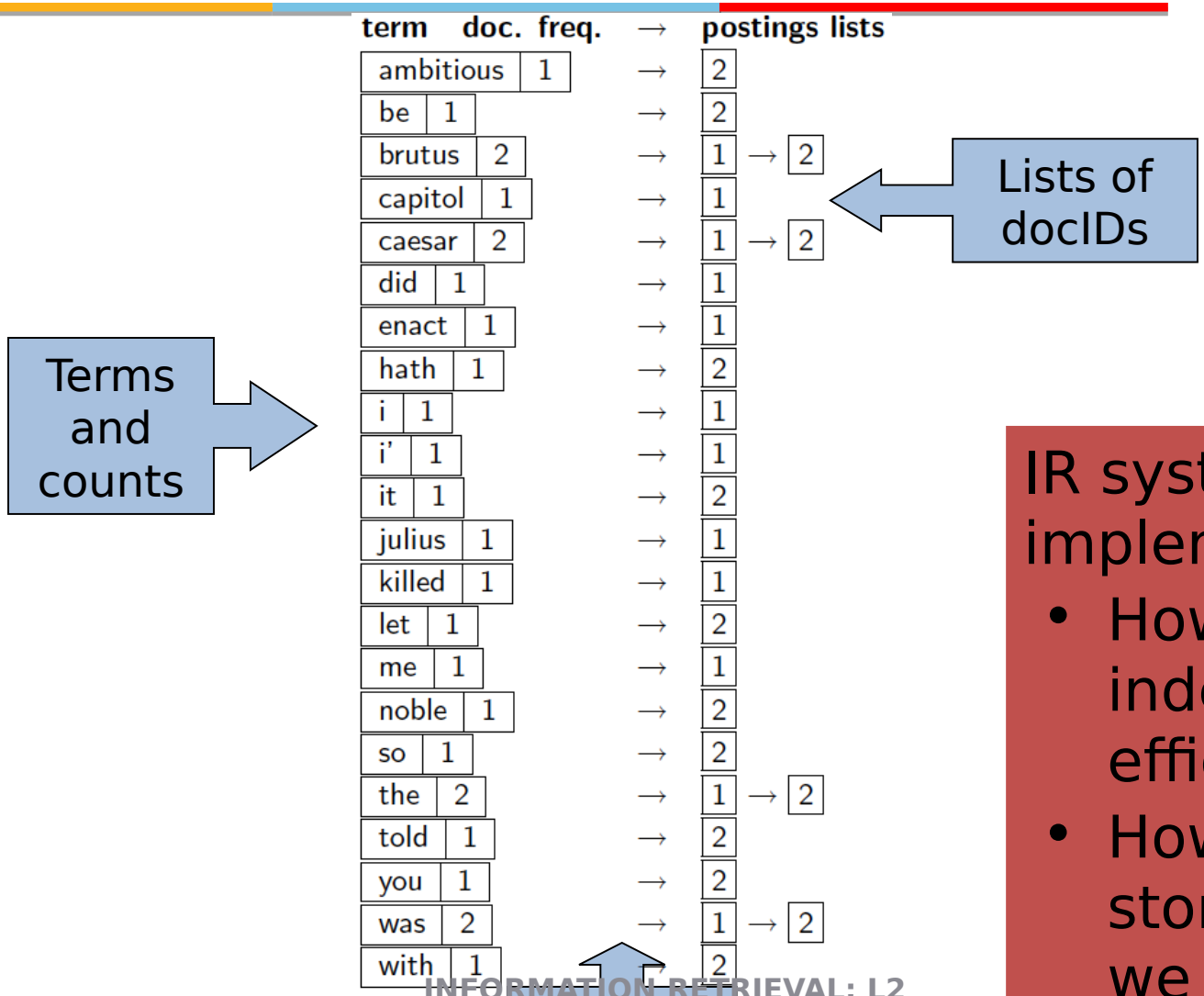
- Terms
- docID

Building Inverted Index

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Document frequency information is added.

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Building Inverted Index


term doc. freq. → postings lists
ambitious 1 → 2
be 1 → 2
brutus 2 → 1 → 2
capitol 1 → 1
caesar 2 → 1 → 2
did 1 → 1
enact 1 → 1
hath 1 → 2
i 1 → 1
i' 1 → 1
it 1 → 2
julius 1 → 1
killed 1 → 1
let 1 → 2
me 1 → 1
noble 1 → 2
so 1 → 2
the 2 → 1 → 2
told 1 → 2
you 1 → 2
was 2 → 1 → 2
with 1 → 2
Terms and counts
Lists of docIDs
Pointers

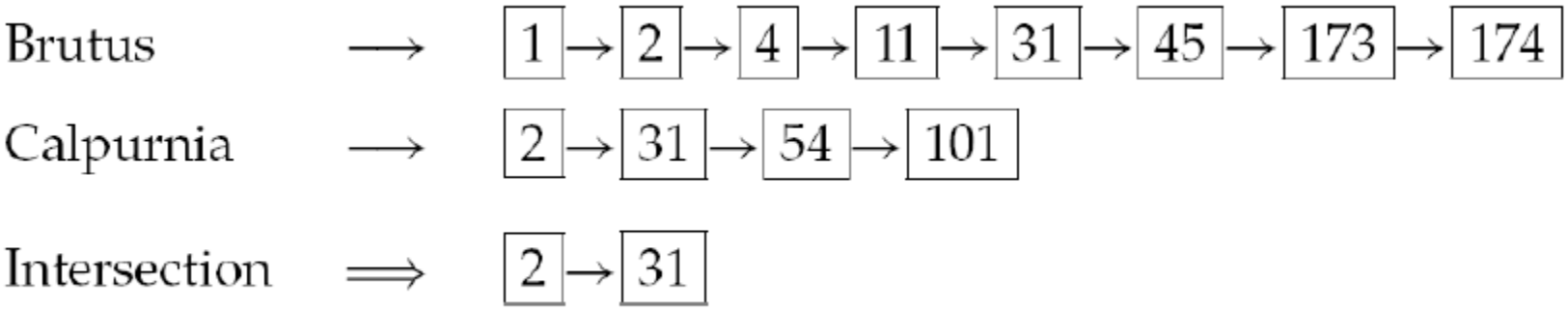
IR system implementation

- How do we index efficiently?
- How much storage do we need?

Processing Boolean Queries

Consider the query: ***Brutus*** *AND* ***Calpurnia***

- Locate **Brutus** in the Dictionary
- Retrieve its postings
- Locate **Calpurnia** in the Dictionary
- Retrieve its postings
- Intersect the two postings lists



Intersecting two postings lists (a "merge" algorithm)

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Time taken for Intersection?

Boolean Query : Exercise Problem

innovate achieve lead

France −→ 1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 11 → 12 → 13 → 14 → 15

Paris −→ 2 → 6 → 10 → 12 → 14

Lear −→ 12 → 15

Compute hit list for ((paris AND NOT france) OR lear)

Boolean Model

1 AND (Intersection)

Narrows your search.

Query: machine AND learning

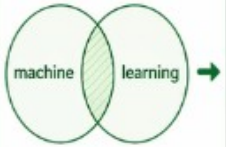
Logic: The system looks up the posting list for "machine" and the posting list for "learning". It then finds the intersection—only the document IDs that appear in both lists.

Inverted Index (Posting Lists)

machine					
1	2	3	5	7	9

learning					
2	3	4	5	8	9

Set Operation



Intersection (AND)

Result

Documents containing BOTH terms

2	3	5	9

Only these documents contain both "machine" AND "learning".

2 OR (Union)

Broadens your search.

Query: AI OR artificial

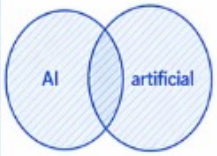
Logic: The system retrieves the posting lists for both terms and merges them, removing duplicates. Any document containing either (or both) terms is returned.

Inverted Index (Posting Lists)

AI				
1	2	4	6	8

artificial					
2	3	4	7	8	10

Set Operation



Union (OR)

Result

Documents containing EITHER term (or both)

1	2	3	4	6	7	8	10

Any document that contains "AI", "artificial", or both.

3 NOT (Difference)

Excludes specific concepts.

Query: apple NOT fruit

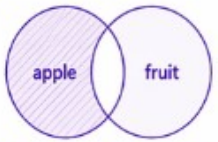
Logic: The system takes the posting list for "apple" and removes any document IDs that also appear in the posting list for "fruit".

Inverted Index (Posting Lists)

apple							
1	2	3′	4	6	8	9	10

fruit					
2	3	5	6	7	10

Set Operation



Difference (apple – fruit)

Result

Documents containing "apple" EXCLUDING "fruit"

1	4	8	9

Documents with "apple" but not "fruit".

Key Takeaway

Boolean operators combine posting lists using set operations:

AND = Intersection (∩) | OR = Union (∪) | NOT = Difference (−)

- Narrows (less results)
- Broadens (more results)
- Excludes (filters out)

Why it matters:

These operators let users express precise information needs using simple, powerful logic.

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using *AND, OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not.
  - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades.
- Many search systems you still use are Boolean:
  - Email, library catalog, macOS Spotlight

Extended Boolean Retrieval Model

Commercially successful Boolean Retrieval: Westlaw

- Largest commercial legal search service in terms of the number of paying subscribers
- Over half a million subscribers performing millions of searches a day over tens of terabytes of text data
- The service was started in 1975.
- In 2005, Boolean search (called “Terms and Connectors” by

  Westlaw) was still the default, and used by a large percentage of users, though ranked retrieval has been available since 1992

Westlaw: Example queries

- **Information need**: What is the statute of limitations in cases involving the federal tort claims act?
- Query:

  LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
- Proximity operators: /3 = within 3 words, /s = within a sentence, /p = within a paragraph
- SPACE is disjunction, not conjunction
- Long, precise queries; proximity operators; incrementally developed; not like web search
- Many professional searchers still like Boolean search
  - You know exactly what you are getting

Westlaw: Example queries

- **Information need**: Information on the legal theories involved in preventing the disclosure of trade secrets by employees formerly employed by a competing company
- Query: “trade secret” /s disclos! /s prevent /s employe!

Westlaw: Example queries

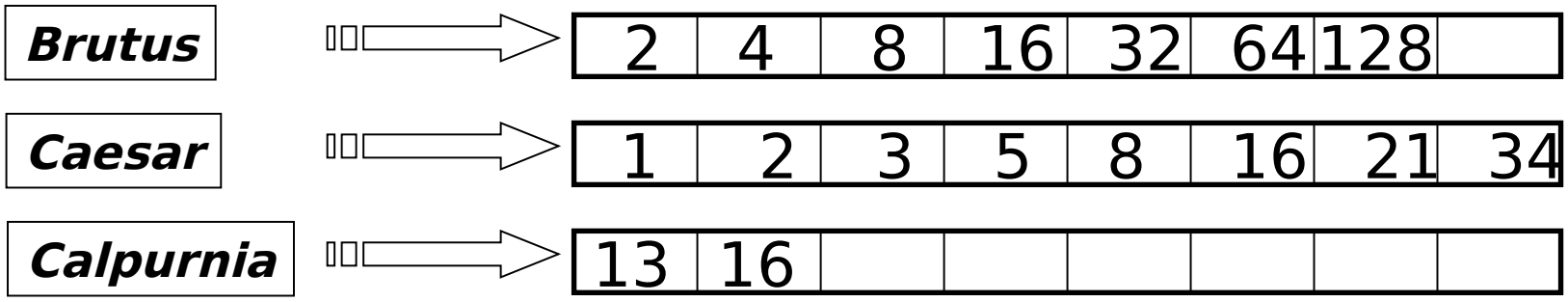
Exercise 1.12

Write a query using Westlaw syntax which would find any of the words professor, teacher, or lecturer in the same sentence as a form of the verb explain.

Query optimization

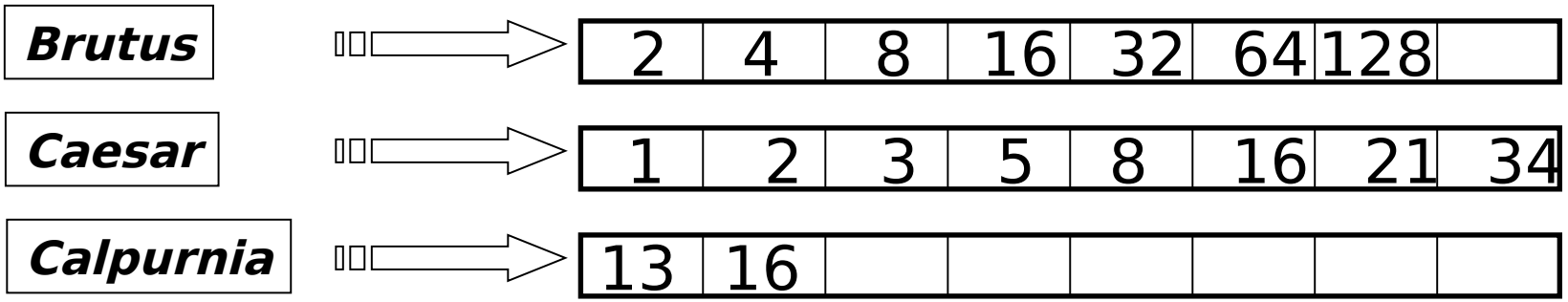
- Consider a query that is an *AND* of *n* terms.
- For each of the *n* terms, get its postings, then *AND* them together.

Query: ***Brutus*** *AND* ***Caesar*** *AND* ***Calpurnia***



Query optimization

- Process in order of increasing frequency:
  - *start with smallest set, then keep cutting further*.



xecute the query as (***Calpurnia*** *AND* ***Brutus)*** *AND* ***Caesar***.

Query optimization

- *(**wind** OR **fire**) AND (**thunder** OR **lightning**)*
- Get doc. frequencies for all terms
- Estimate the size of each *OR* by the sum of its doc. freq.'s
- Process in increasing order of *OR* sizes.

Example Problem 1

innovate achieve lead

Consider these documents:

Doc 1 : breakthrough drug for schizophrenia

Doc 2 : new schizophrenia drug

Doc 3 : new approach for treatment of schizophrenia

Doc 4 : new hopes for schizophrenia patients

a. **Draw the term-document incidence matrix for this document collection.**

b. **Draw the inverted index representation for this collection.**

Example Problem 2

innovate achieve lead

- Recommend a query processing order for

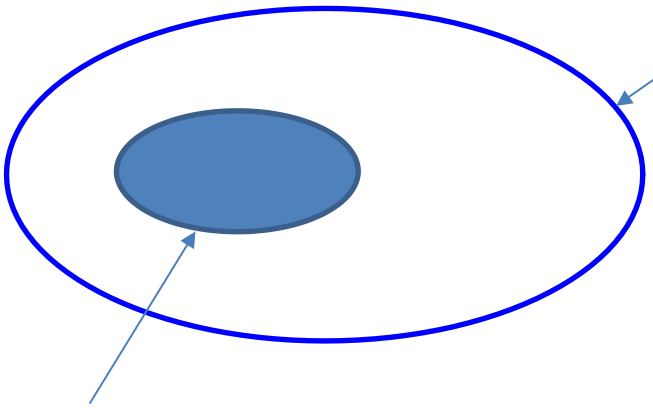
(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes)

given the following postings list sizes:

Term	Postings size
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Boolean Model- Evaluation

- Documents retrieved are **Relevant** to the information need
  - All relevant documents are retrieved?

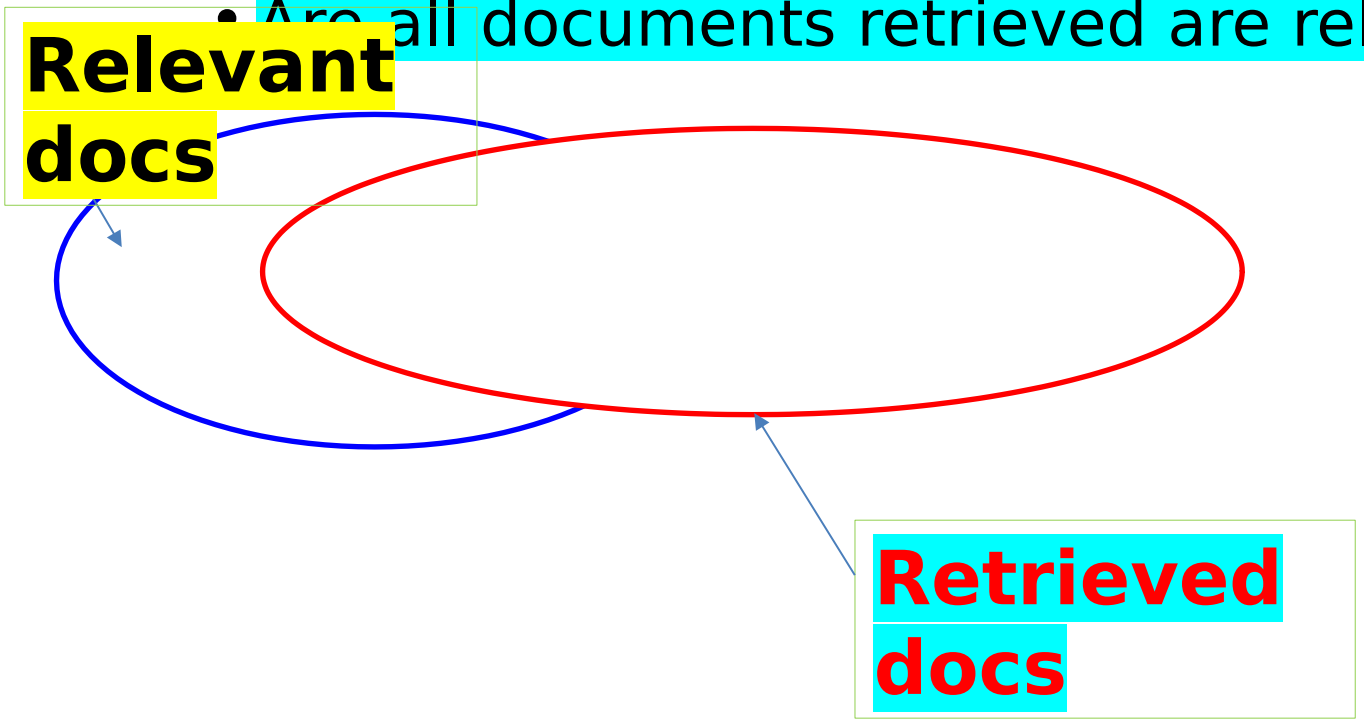


Relevant docs

Retrieved docs

Boolean Model- Evaluation

- Documents retrieved are **Relevant** to the information need
  - Are all documents retrieved are relevant?



Boolean Model- Evaluation

Evaluation Measures

- **Precision**
  - What fraction of the returned results are relevant to the information need?
- **Recall**
  - What fraction of the relevant documents in the collection were returned by the system?

If A is the number of Relevant docs in the collection, C is the total documents retrieved and B is the relevant docs retrieved.

- Precision =
- Recall =

Boolean Model- Evaluation

Evaluation Measures

- **Precision**
- What fraction of the returned results are relevant to the information need?
- **Recall**
- What fraction of the relevant documents in the collection were returned by the system?

If A is the number of Relevant docs in the collection, C is the total documents retrieved and B is the relevant docs retrieved.

- Precision = B/C
- Recall = B/A

Example Problem 3

innovate achieve lead

- Consider the four terms, in order: park, mountain, trails, difficult. Assume that the query, in disjunctive normal form, DNF, is the following, where "OR" is the logical disjunction operator:

Q = (1,0,1,0) OR (0,1,1, 0) OR (1,1,1,0)

Can you give the description of the query ?

Document 1: "Loon park contains a lovely lake and is near Mystery mountain. It's not difficult to get to from the city."

Document 2: "The Mystery mountain area has many easy trails, but no difficult ones.“

- What will be result of the query when applied to these documents?

Resources

https://www.ajer.org/papers/v6(11)/ZA0611216222.pdf

Efficiency of Boolean Search strings for Information Retrieval

Muhammad Bello Aliyu

Computer Science Unit, Department of Mathematics, Usmanu Danfodiyo University Sokoto

American Journal of Engineering Research (AJER)